

JCC10 U.S. PTO
08/18/00

8-21-00 A

Please type a plus sign (+) inside this box → PTO/SB/05 (4/98)
Approved for use through 09/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

**UTILITY
PATENT APPLICATION
TRANSMITTAL**

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No. 103.1033.01

First Inventor or Application Identifier Blake Lewis

Title Reserving File System Blocks

Express Mail Label No. EL 524 781 089 US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. <input type="checkbox"/> * Fee Transmittal Form (e.g., PTO/SB/17) (Submit an original and a duplicate for fee processing)	5. <input type="checkbox"/> Microfiche Computer Program (Appendix)
2. <input checked="" type="checkbox"/> Specification [Total Pages 23] (preferred arrangement set forth below)	6. Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)
- Descriptive title of the Invention	a. <input type="checkbox"/> Computer Readable Copy
- Cross References to Related Applications	b. <input type="checkbox"/> Paper Copy (identical to computer copy)
- Statement Regarding Fed sponsored R & D	c. <input type="checkbox"/> Statement verifying identity of above copies
- Reference to Microfiche Appendix	
- Background of the Invention	
- Brief Summary of the Invention	
- Brief Description of the Drawings (if filed)	
- Detailed Description	
- Claim(s)	
- Abstract of the Disclosure	
3. <input checked="" type="checkbox"/> Drawing(s) (35 U.S.C. 113) [Total Sheets 5]	
4. Oath or Declaration [Total Pages]	
a. <input type="checkbox"/> Newly executed (original or copy)	7. <input type="checkbox"/> Assignment Papers (cover sheet & document(s))
b. <input type="checkbox"/> Copy from a prior application (37 C.F.R. § 1.63(d)) (for continuation/divisional with Box 16 completed)	8. <input type="checkbox"/> 37 C.F.R. § 3.73(b) Statement <input type="checkbox"/> Power of (when there is an assignee) <input type="checkbox"/> Attorney
i. <input type="checkbox"/> DELETION OF INVENTOR(S)	9. <input type="checkbox"/> English Translation Document (if applicable)
Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).	10. <input type="checkbox"/> Information Disclosure Statement (IDS)/PTO-1449 <input type="checkbox"/> Copies of IDS Citations
	11. <input type="checkbox"/> Preliminary Amendment
	12. <input checked="" type="checkbox"/> Return Receipt Postcard (MPEP 503) (Should be specifically itemized)
	* Small Entity Statement(s) <input type="checkbox"/> Statement filed in prior application, (PTO/SB/09-12) <input type="checkbox"/> Status still proper and desired
	13. <input type="checkbox"/> Certified Copy of Priority Document(s) (if foreign priority is claimed)
	14. <input type="checkbox"/> Other: certificate of mailing
	15. <input checked="" type="checkbox"/> Other: _____

NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).

16. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment:

Continuation Divisional Continuation-in-part (CIP) of prior application No: _____

Prior application information: Examiner _____ Group / Art Unit: _____

For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

Customer Number or Bar Code Label **22883** or Correspondence address below
(Insert Customer No. or Attach bar code label here)

Name	PATENT TRADEMARK OFFICE		
Address			
City	State	Zip Code	
Country	Telephone	Fax	

Name (Print/Type)	Steven A. Swernofsky	Registration No. (Attorney/Agent)	33,040
Signature	<i>Steven A. Swernofsky</i>	Date	August 18, 2000

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

USPS Pro
103/642063
09/18/00

Certificate of Mailing (37 C.F.R. § 1.10)

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the United States Postal Services on the date shown below as "Express Mail" (Post Office to Addressee) in an envelope addressed to the Honorable Assistant Commissioner for Patents and Trademarks, Box Patent Application, Washington, D.C. 20231.

Mailing Label No. EL 524 781 089 US

Date of Deposit: August 18, 2000

Arlette Malhas

Printed Name

Arlette Malhas

Signature

Documents enclosed:

- Patent Application Transmittal Form;
- Specification (17) pages;
- Claims (5) pages;
- Abstract (1) pages;
- Drawings (5) pages;
- Return post card; and
- Certificate of Express Mailing.

1 This application is submitted in the name of the following inventors:
2

3 Inventor Citizenship Residence City and State
4 Blake LEWIS United States Palo Alto, California
5 Kayuri PATEL Kenya Cupertino, California
6 Ray CHEN United States Campbell, California
7

8 The assignee is Network Appliance, Inc., a California corporation having an office
9 at 495 East Java Dr., Sunnyvale, CA 94089.
10

11 Title of the Invention
12

13 Reserving File System Blocks
14

15 Background of the Invention
16

17 *1. Field of the Invention*
18

19 This invention relates to management of a file system for a file server. In
20 particular, the invention concerns reserving unallocated blocks of the file system based upon a
21 file size for a file.
22

23 *2. Description of the Related Art*
24

25 Some conventional file servers (also called “filers”) manage space in a file system
26 by allocating blocks to a file as data for that file is written to the file system. Thus, if the file
27 system runs out of space, a file could end up partially written.
28

1 Other conventional file servers, in particular those running CIFS, can allocate
2 blocks for an entire file upon creation of the file. These file servers write zero data to the file
3 system for these blocks. However, this approach is expensive in terms of time required to write
4 the zero data. Furthermore, this approach is actually counterproductive for file servers that use a
5 write anywhere file system layout, also known as WAFL file servers.

6

7 In WAFL file servers, when a file is overwritten, new data is written to new
8 blocks, and then the previously allocated blocks are released. Thus, new data is written to
9 different blocks than the previously allocated blocks, resulting in use of extra space for the new
10 blocks until the previously allocated blocks are released. If the file server is close to full, this
11 duplication of blocks could use up the remaining blocks, preventing complete writing of the data.

12 //
13 //
14 //
15 //

1 Summary of the Invention

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

Accordingly, what is needed is a way to ensure that enough blocks are reserved for a file so as to ensure that the entire file can be written to a file system, without actually allocating disk blocks to the file.

In one aspect, the invention addresses the foregoing need through a method of managing a file system for a file server. According to the method, a file operation is received that signals a reservation operation for a file having a file size. Preferably, the file system uses a write anywhere file system layout, the file operation that signals the reservation operation is a zero length write request, and the file operation that signals the reservation operation includes a parameter that specifies the file size. A number of blocks needed to be reserved to accommodate the file is computed. Preferably, computing the number of blocks needed to be reserved to accommodate the file includes determining a total number of direct and indirect blocks needed to accommodate the file size, and subtracting a total number of blocks already allocated for the file and a total number of cached unallocated blocks for the file from the total number of direct and indirect blocks needed to accommodate the file size. Unallocated blocks are reserved in the file system, with the number of reserved blocks equal to the number of blocks needed to be reserved to accommodate the file. Reserving the number of blocks preferably includes setting a flag in an inode for the file that indicates blocks have been reserved for the file, and incrementing a reserved block count in a file system information block by the number of blocks needed. The reserved block count indicates how many unallocated blocks have been reserved for files in the file system.

In a preferred embodiment, the method also includes the step of checking that a number of available blocks in the file system is greater than the number of blocks needed to be

1 reserved to accommodate the file. An error is returned in a case that the number of available
2 blocks is less than the number of blocks needed. The number of available blocks in the file
3 system preferably is determined by subtracting a number of allocated blocks, a number of cached
4 unallocated blocks (i.e., delayed allocated blocks), and a number of reserved blocks from a total
5 number of blocks in the file system, and adding to this a number of reserved cached unallocated
6 blocks.

7

8 Also in the preferred embodiment, the file server checks that the number of blocks
9 needed to be reserved to accommodate the file does not exceed a remainder of a quota for an
10 owner of the file. An error is returned in a case that the number of blocks needed exceeds the
11 remainder of the quota.

12

13 When data is written to the file system, blocks are cached in a buffer cache. At a
14 later point in time, the blocks are stored to storage for the file system. Prior to writing the blocks
15 to disk, the blocks are actually allocated to the files. Reservation of those blocks is released as
16 the blocks are written to storage. Releasing reservation of blocks is accomplished by
17 decrementing the reserved block count in the file system information block by a number of
18 released blocks.

19

20 By virtue of the foregoing operations, a file server reserves unallocated blocks for
21 a file for which file reservation semantics are activated. These reserved blocks are not actually
22 allocated by the reservation process. Rather, a count is maintained of how many blocks need to
23 be kept available for the file system. This count is utilized when space availability for the file
24 system is checked, thereby helping to ensure that enough blocks are available for files that have
25 reserved file space.

26

1 In another aspect, the foregoing method handles receipt of a file operation that
2 signals a reservation operation for a file for which reservation has already been performed, in
3 which the reservation operation specifies a new file size different from a current file size for the
4 file. When such a file operation is received, additional blocks may need to be reserved to
5 accommodate the new file. According to the method, a current file size for the file and the new
6 file size are compared. In a case that the current file size exceeds the new file size, remaining
7 block reservations for the file are released. In a case that the new file size exceeds the current file
8 size, an additional number of unallocated blocks are reserved in the file system. According to
9 this embodiment of the invention, the additional number of unallocated blocks to be reserved
10 equals a difference between a total number of direct and indirect blocks required by the new file
11 size and a total number of direct and indirect blocks required by the current file size.

12 By virtue of the foregoing operation, changes in file size for a file can be
13 appropriately reflected in the block reservations.

14 Each of the foregoing methods can be used in conjunction with the others in
15 various combinations to perform reservation operations. The invention also can be embodied in
16 apparatuses such as file servers and/or other hardware configured to perform the foregoing
17 methods, computer readable code by itself or embodied in a computer program product to cause a
18 computer to perform the foregoing methods, and a memory storing information including
19 instructions executable by a processor to perform the foregoing methods.

20 This brief summary has been provided so that the nature of the invention may be
21 understood quickly. A more complete understanding of the invention may be obtained by
22 reference to the following description of the preferred embodiments thereof in connection with
23 the attached drawings.

1 Brief Description of the Drawings

2

3

4 Figure 1 shows an environment in which a file server manages a file system

5

6

7 Figure 2 is a block diagram of an organization of a file system in which blocks can

8

9

10 Figure 3 is a flowchart for explaining reservation according to the invention of

11

12

13

14

15

16

17

18

19

20

21

22

23

24

Figure 3 is a flowchart for explaining reservation according to the invention of
blocks for a file.

Figure 4 is a flowchart for explaining computation of a number of blocks needed

for a file.

Figure 5 is a flowchart for explaining determination of how many blocks are

available for a file system.

Figure 6 is a flowchart for explaining reservation of unallocated blocks for a file.

Figure 7 is a flowchart for explaining allocation and writing of data to a file in a

file system in which blocks can be reserved for files.

Figure 8 is a flowchart for explaining block reservation when a file size changes.

1 Description of the Preferred Embodiment

2
3 In the following description, a preferred embodiment of the invention is described
4 with regard to preferred process steps and data structures. However, those skilled in the art
5 would recognize, after perusal of this application, that embodiments of the invention may be
6 implemented using one or more general purpose processors or special purpose processors
7 adapted to particular process steps and data structures operating under program control, that such
8 process steps and data structures can be embodied as information stored in or transmitted to and
9 from memories (e.g., fixed memories such as DRAMs, SRAMs, hard disks, caches, etc., and
10 removable memories such as floppy disks, CD-ROMs, data tapes, etc.) including instructions
11 executable by such processors (e.g., object code that is directly executable, source code that is
12 executable after compilation, code that is executable through interpretation, etc.), and that
13 implementation of the preferred process steps and data structures described herein using such
14 equipment would not require undue experimentation or further invention.

15
16 Fig. 1 shows an environment in which a file server manages a file system
17 according to the invention. In Fig. 1, clients 1 send data to and retrieve data from server 2, which
18 stores the data in file system 3. File system 3 includes storage 4 such as multiple hard disk
19 drives, multiple optical drives, or any other mass storage. Information is stored in storage 4 in
20 blocks, as explained in more detail below with reference to Fig. 2. Preferably, file system 3 uses
21 a write anywhere file system layout (WAFL).

22 When data is sent to server 2, that data is initially stored in buffer cache 5 before
23 being written to file system 3. Data in buffer cache 5 also is stored in blocks, which are shown as
24 cached blocks 6 in Fig. 1. According to the invention, each cached block 6 preferably includes
25 flag 7 that indicates whether or not the block was cached in buffer cache 5 after block reservation
26 according to the invention was activated for the file to which the block belongs. The operation of

1 flag 7 is discussed below with respect to Figs. 3 to 8.

2

3 Blocks in buffer cache 5 can be copies of allocated blocks already written to
4 storage 4, or unallocated blocks which have not yet been written to storage 4. An allocated block
5 has a block identification number (not shown) associated therewith that indicates to which block
6 of storage 4 it corresponds. An unallocated blocks has no such block identification number
7 associated therewith, or has a block identification number of zero.

8

9 When a new block is created in buffer cache 5, that block is “clean.” When data
10 is written to the block, the block is “dirtied.” File server 2 preferably uses delayed allocation for
11 these dirty blocks. According to delayed allocation, blocks in storage 4 are not immediately
12 allocated for the dirty blocks, and the dirty blocks preferably are not immediately written to
13 storage 4. Thus, the dirty blocks are cached unallocated blocks. Such cached unallocated blocks
14 are also called delayed allocated blocks.

15

16 Actual writing of the cached unallocated blocks (i.e., delayed allocated blocks)
17 occurs at certain predetermined times or after the occurrence of certain predetermined conditions
18 (i.e., NVRAM for persistent storage of file server operations fills up, a preset number of buffers
19 or blocks are cached, etc.). Then, blocks in storage 4 are allocated for dirtied blocks in buffer
20 cache 5, and the dirtied blocks are actually stored in storage 4.

21

22 File server 2 also includes incore information structure 8 with incore inodes 9 and
23 incore information 10. An incore inode 9 is present for each file for which blocks are stored in
24 buffer cache 5. The incore inodes associate the blocks with the file in a manner similar to inode
25 14 discussed below with respect to Fig. 2.

26

1 Each incore inode also includes a count of the number of cached unallocated
2 blocks (i.e., delayed allocated blocks) associated with the inode. Incore information 10
3 preferably includes a count of how many total cached unallocated blocks are stored in buffer
4 cache 5.

5
6 Fig. 2 is a block diagram of an organization of file system 3. For the sake of
7 brevity, many of the conventional features of file system 3 are not illustrated in Fig. 2 and are not
8 discussed herein. Those skilled in the art will appreciate that a great many ways exist to
9 implement these conventional features without departing from the invention.
10
11

12 File system 3 includes at least one system information block 11. This system
13 information block includes at least block allocation information 12 and reserved block count 13
14 according to the invention. Block allocation information 12 includes at least information from
15 which can be determined how many blocks of file system 3 are allocated to files. Reserved block
16 count 13 includes a count of how many blocks in file system 3 have been reserved according to
17 the invention.

18 An inode such as inode 14 is associated with each file stored in file system 3.
19 Inode 14 is stored in storage 4. Inode 14 includes at least flag 15 that indicates whether or not
20 block reservation according to the invention is active for the file to which the inode belongs. The
21 operation of flag 15 is discussed below with respect to Figs. 3 to 8. Inode 14 also can include file
22 size information for its associated file.

23
24 Inode 14 for a file further includes information associating that file with blocks 16
25 in file system 3. These blocks preferably are 4 kilobytes long. Blocks 16 can be direct blocks, in
26 which case data for the file is stored directly in the blocks. Blocks 16 also can be indirect blocks

1 that store information associating other blocks with the file. Those other blocks in turn can store
2 data or can point to yet more blocks.

3

4 For a file size of less than 64 bytes, no blocks are needed. Instead, data for the file
5 is stored directly in inode 14. In a typical block configuration, each inode can point to up to
6 sixteen other blocks. Thus, for file sizes between 64 bytes and 64 kilobytes, up to sixteen direct
7 blocks are utilized. Each block in turn can point to 1024 other blocks. Thus, one level of
8 indirection can accommodate files between 64 kilobytes and 64 megabytes (16 blocks pointing to
9 1024 blocks of 4 kilobytes each). Two levels of indirection can accommodate file sizes up to 64
10 gigabytes (16 blocks pointing to 1024 blocks, which each in turn point to 1024 blocks of 4
11 kilobytes each). More levels of indirection can be utilized, as needed. Of course, the invention is
12 equally applicable to file systems that utilize different size blocks than 4 kilobytes and that are
13 organized in different arrangements than shown in Fig. 2

14

15 As data is written to a file in file system 3, the data is cached in block 6 in the
16 buffer cache. Prior to writing the block 6 to storage 4, blocks are allocated to that file. File server
17 2 manages the storage of the data in the blocks and keeps track of block allocation. According to
18 the invention, file server 2 also can reserve blocks for files before data is written to those files.
19 File server 2 uses reserved block count 13 to keep track of how many blocks are reserved.

20

21 Reserved blocks preferably are not actually assigned to a file. Instead, file server
22 2 ensures that at least the number of blocks indicated by reserved block count 13 are kept free.
23 Then as data is written to a file for which reservation is activated, the data is cached in block 6 in
24 the buffer cache. Prior to writing the block 6 to storage 4, blocks are allocated to that file.
25 Because those blocks are now allocated, space for the blocks no longer needs to be reserved in
26 file system 3, so reserved block count 13 is decremented accordingly. These operations are

1 discussed in more detail below.

2

3 Block allocation information 12, reserved block count 13, and inodes 14 with
4 flags 15 preferably are stored in file system 3 on storage 4. Thus, whenever a snapshot of file
5 system 3 is made for backup purposes, sufficient block allocation and reservation information is
6 stored in the snapshot to reconstruct the block allocation and reservation status from the
7 snapshot.

8

9 Fig. 3 is a flowchart for explaining reservation according to the invention of
10 blocks for a file.

11

12 Briefly, the invention manages a file system for a file server. A file operation is
13 received that signals a reservation operation for a file having a file size. A number of blocks that
14 need to be reserved to accommodate the file is computed. A number of unallocated blocks is
15 reserved in the file system equal to the number of blocks needed to be reserved to accommodate
16 the file.

17

18 In more detail, in step S301, file server 2 receives a file operation that signals a
19 reservation operation for a file. In the preferred embodiment of the invention, this file operation
20 is a zero length write request. This write request preferably includes a parameter that identifies a
21 size for the file for which the reservation is to be made. Alternatively, the file size can already be
22 set in file system 3, possibly through a previous write command.

23

24 In step S302, a number of blocks needed to be reserved to accommodate the file is
25 computed. This step is shown in more detail in Fig. 4.

26

1 In step S401 of Fig. 4, a number of blocks needed to accommodate the file size is
2 computed. This number preferably is irrespective of information such as actual block allocation
3 and reservation data. The number of blocks includes both direct blocks and indirect blocks, as
4 described above with respect to Fig. 2.

5

6 For example, for a file of size 64 kilobytes, a preferred embodiment of the
7 invention needs 16 direct blocks and no indirect blocks, for a total of 16 blocks. The number
8 computed in step S401 preferably is generated by an algorithm that takes a file size as an input
9 and generates a number of blocks as an output.

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

In step S402 of Fig. 4, a number of blocks already allocated to the file, for example as a result of previous write operations, is subtracted from the number of blocks needed to accommodate the file size. Information about the number of blocks allocated to the file preferably is retrieved or derived from block allocation information 12 and/or inodes 14.

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

3310

3311

3312

3313

3314

3315

3316

3317

3318

3319

3310

3311

3312

3313

3314

3315

3316

3317</

1 steps S402 and S403 can occur simultaneously or in a different order, as long as the number of
2 needed blocks is determined.

3

4 Returning to Fig. 3, the number of blocks available for file system 3 is determined
5 in step S303. This step is shown in more detail in Fig. 5.

6

7 In step S501 of Fig. 5, a total number of blocks for file system 3 is determined.
8 This number is a substantially fixed number that usually only changes when storage 4 is
9 physically altered, for example by permanent corruption of blocks (i.e., bad blocks), changes in
hardware (i.e., addition or removal of disk drives), reformatting, and the like.

10 In step S502, a number of allocated blocks is subtracted from the total number of
blocks. The number of allocated blocks preferably is determined from block allocation
information 12 in system information block 11 of file system 3.

11

12 In step S503, a number of cached unallocated blocks (i.e., delayed allocated
blocks) is subtracted from the result of step S503. The number of delayed allocated blocks is
determined from incore information 10 for file system 3.

13

14 In step S504, a number of reserved blocks is subtracted from the result of step
15 S503. The number of reserved blocks preferably is determined from reserved block count 13 in
16 system information block 11 of file system 3.

17

18 In step S505, a number of reserved cached unallocated blocks is added to the
result of step S504. File server 2 preferably keeps track of how many cached unallocated blocks
in buffer cache 5 have flag 7 set, thereby indicating that those blocks are blocks for which
reservation according to the invention is activated. This information preferably is kept up to date

1 in incore information 10 for file system 3. This step is necessary since the reserved cached
2 unallocated blocks are accounted for in the delayed allocation counters in incore information 10,
3 since actual blocks in storage 4 have not yet been allocated for these blocks.

4

5 The result of steps S501 through S505 is a number of available blocks for the file
6 system. Of course, steps S501 through S505 can occur simultaneously or in a different order, as
7 long as the number of available blocks is determined.

8

9 Returning to Fig. 3, in step S304, the number of blocks needed to be reserved is
10 compare with the number of available blocks. If the number of needed blocks exceeds the
11 number of available blocks, not enough space is available in file system 3, and flow is diverted to
12 step S305, where an error is returned. Otherwise, flow proceeds to step S306.

13

14 In step S306, a number of blocks remaining in the file owner's quota is
15 determined. This operation is only performed if quotas are being enforced for file system 3.
16 Then, in step S307, the quota is compared against the number of blocks needed to be reserved. If
17 not enough blocks remain in the quota, flow is diverted to step S308, where an error is returned.
18 Otherwise, flow proceeds to step S309.

19

20 Unallocated blocks are reserved in step S309. The number of blocks reserved is
21 equal to the number of blocks needed, as determined in step S302. Step S309 is shown in more
22 detail in Fig. 6.

23

24 In step S601, flag 15 in inode 14 for the file is set. This flag indicates that
25 reservation semantics are being enforced for the file. Then, in step S602, reserved block count
26 13 in system information block 11 is incremented by the number of blocks needed to be reserved

1 for the file.

2
3 After step S309, blocks have been reserved for the file.

4
5 Fig. 7 is a flowchart for explaining allocation and writing of data to a file in a file
6 system in which blocks can be reserved for files.

7
8 When data is to be written to file system 3 for a file, step S701 checks flag 15 of
9 inode 14 for that file to determine whether or not block reservation has been performed for the
10 file. If reservation has been performed, flow proceeds to step S705. Otherwise, flow proceeds to
11 step S702 for space availability checking.

12
13 The number of blocks available for file system 3 is determined in step S702.
14 Preferably, step S702 subtracts a number of allocated blocks, a number of cached unallocated
15 blocks (i.e., delayed allocated blocks), and a number of reserved blocks from a total number of
16 blocks in the file system, and adds to this a number of reserved cached unallocated blocks.

17
18 In step S703, the number of blocks needed to be allocated is compared with the
19 number of available blocks. If the number of needed blocks exceeds the number of available
20 blocks, not enough space is available in file system 3, and flow is diverted to step S704 where an
21 error is returned. Otherwise, flow proceeds to steps S705.

22
23 In step S705, the data is written to a block or blocks in buffer cache 5 for file
24 system 3. If reservation has been activated for the file, file server 3 ensures that flags 7 are set for
25 the blocks.

26

1 Flow proceeds from step S705 to step S706 in order to write data from buffer
2 cache 5 to storage 4. As mentioned above, data preferably is written from the buffer cache to the
3 storage after the occurrence of certain predetermined conditions (i.e., NVRAM for persistent
4 storage of file server operations fills up, a preset number of buffers or blocks are cached, etc.).
5

6 In step S706, file server 3 checks flags 7 for each block to be written to storage 4.
7 File server 3 checks flags 7 to see if reservation according to the invention has been activated for
8 the blocks. If reservation has been activated, flow proceeds to step S707. Otherwise, flow skips
9 to step S708.

10
11 In step S707, reserved block count 13 is decremented by the number of reserved
12 blocks to be written to storage. Because these blocks are actually going to be written to storage
13 4, space for the blocks no longer needs to be reserved. Decrementing the reserved block count
14 releases the reservation of these blocks.

15 In step S708, a block or blocks of storage 4 are allocated for the data, and in step
16 S709, the data is written to the block or blocks of storage 4.
17

18 Fig. 8 is a flowchart for explaining a reservation operation for a file for which
19 reservation has already been performed, in which the reservation operation specifies a new file
20 size different from a current (old) file size for the file. For example, the steps in Fig. 8 are
21 performed when a first zero length write specifies a first file size, and then a second zero length
22 write specifies a second different file size.
23

24 In step S801, the current file size is compared with a new file size for the
25 reservation command. If the new file size is less than the old file size, flow proceeds to step
26 S802, where remaining block reservations for the file are released. In order to release the block

1 reservations, flag 15 in inode 14 for the file is cleared, and reserved block count 13 is
2 decremented by the number of blocks still reserved for the file. This number is computed along
3 the lines of step S302 shown in Figs. 3 and 4, based on the current file size, the number of blocks
4 allocated to the file, and the number of cached unallocated blocks for the file.

5

6 If the new file size is greater than the old file size, flow proceeds to step S803,
7 where more blocks are reserved. The number of additional blocks reserved preferably is equal to
8 the difference between the total number of direct and indirect blocks required by the new file size
9 and the total number of direct and indirect blocks required by the current file size. These
10 additional blocks are reserved by incrementing reserved block count 13 by this difference.

11 *Alternative Embodiments*

12

13 Although preferred embodiments of the invention are disclosed herein, many
14 variations are possible which remain within the content, scope and spirit of the invention, and
15 these variations would become clear to those skilled in the art after perusal of this application.
16 For example, the invention is equally applicable to file servers and file systems that use different
17 block sizes and block organizations, different buffer cache schemes (e.g., no delayed allocation),
18 different layouts than the WAFL layout, and the like. In addition, the invention is not limited to
19 the particular orderings of steps described above. Therefore, the scope of the invention
20 encompasses the following claims and their legal equivalents and is not limited to the
21 embodiment discussed above.

22 //

23 //

24 //

25 //

26 //

1 Claims

2

3 What is claims is:

4

5 1. A method of managing a file system for a file server, comprising the steps of:
6 receiving a file operation that signals a reservation operation for a file having a
7 file size;
8 computing a number of blocks needed to be reserved to accommodate the file; and
9 reserving a number unallocated blocks in the file system equal to the number of
10 blocks needed to be reserved to accommodate the file.

11

12 2. A method as in claim 1, wherein the file system uses a write anywhere file
13 system layout.

14

15 3. A method as in claim 1, wherein the file operation that signals the reservation
16 operation is a zero length write request.

17

18 4. A method as in claim 1, wherein the file operation that signals the reservation
19 operation includes a parameter that specifies the file size.

20

21 5. A method as in claim 1, wherein the step of computing the number of blocks
22 needed to be reserved to accommodate the file further comprises:

23 determining a total number of direct and indirect blocks needed to accommodate
24 the file size; and

25 subtracting a total number of blocks already allocated for the file and a total
26 number of cached unallocated blocks for the file from the total number of direct and indirect

1 blocks needed to accommodate the file size.

2

3 6. A method as in claim 1, wherein the step of reserving the number of
4 unallocated blocks in the file system equal to the number of blocks needed further comprises:
5 setting a flag in an inode for the file that indicates blocks have been reserved for
6 the file; and

7 incrementing a reserved block count in a file system information block by the
8 number of blocks needed, the reserved block count indicating how many unallocated blocks have
9 been reserved for files in the file system.

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

7. A according to claim 1, further comprising the step of checking that a number
of available blocks in the file system is greater than the number of blocks needed to be reserved
to accommodate the file, wherein an error is returned in a case that the number of available
blocks is less than the number of blocks needed.

8. A method as in claim 7, wherein the number of available blocks in the file
system is determined by subtracting a number of allocated blocks, a number of cached
unallocated blocks, and a number of reserved blocks from a total number of blocks in the file
system, and adding a number of reserved cached unallocated blocks.

9. A method according to claim 1, further comprising the step of checking that
the number of blocks needed to be reserved to accommodate the file does not exceed a remainder
of a quota for an owner of the file, wherein an error is returned in a case that the number of
blocks needed exceeds the remainder of the quota.

10. A method as in claim 1, further comprising the step of releasing reservation

1 of blocks as those blocks are written to storage.

2

3 11. A method as in claim 10, wherein the step of releasing reservation of blocks
4 further comprises the step of decrementing a reserved block count in a file system information
5 block by a number of released blocks, the reserved block count indicating how many unallocated
6 blocks have been reserved for files in the file server.

7

8 12. A method of managing a file system for a file server, comprising the steps of:
9 receiving a file operation that signals a reservation operation for a file for which
10 reservation has already been performed, said reservation operation specifying a new file size
11 different from a current file size for the file;

12 comparing the current file size with the new file size;

13 in the case that the current file size exceeds the new file size, releasing the
14 remaining block reservations for the file;

15 in the case that the new file size exceeds the current file size, reserving in the file
16 system an additional number of unallocated blocks equal to a difference between a total number
17 of direct and indirect blocks required by the new file size and a total number of direct and
18 indirect blocks required by the current file size.

19

20 13. A method as in claim 12, wherein the file system uses a write anywhere file
21 system layout.

22

23 14. A method as in claim 12, wherein the file operation that signals the
24 reservation operation is a zero length write request.

25

26 15. A method as in claim 12, wherein the file operation that signals the

1 reservation operation includes a parameter that specifies the file size.

2

3 16. A method as in claim 12, wherein the step of releasing remaining block
4 reservations for the file further comprises the steps of:

5 resetting a flag in an inode for the file that indicates blocks have been reserved for
6 the file; and

7 decrementing a reserved block count in a file system information block by a
8 number of blocks still reserved for the file, the reserved block count indicating how many
9 unallocated blocks have been reserved for files in the file system.

10

11 17. A method as in claim 12, further comprising the step of checking that a
12 number of available blocks in the file system is greater than the additional number of unallocated
13 blocks, wherein an error is returned in a case that the number of available blocks is less than the
14 additional number of blocks.

15

16 18. A method as in claim 17, wherein the number of available blocks in the file
17 system is determined by subtracting a number of allocated blocks, a number of cached
18 unallocated blocks, and a number of reserved blocks from a total number of blocks in the file
19 system, and adding a number of reserved cached unallocated blocks.

20

21 19. A method according to claim 12, further comprising the step of checking that
22 the additional number of blocks does not exceed a remainder of a quota for an owner of the file,
23 wherein an error is returned in a case that the additional number of blocks exceeds the remainder
24 of the quota.

25

26 20. A method as in claim 12, further comprising the step of releasing reservation

1 of blocks as those blocks are written to storage.

2

3 21. A method as in claim 20, wherein the step of releasing reservation of blocks
4 further comprises the step of decrementing a reserved block count in a file system information
5 block by a number of released blocks, the reserved block count indicating how many blocks total
6 have been reserved for files in the file server.

7 //

8 //

9 //

10 //

11 //

12 //

13 //

14 //

15 //

16 //

17 //

1 Abstract

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

A system that manages a file system for a file server. A file operation is received that signals a reservation operation for a file having a file size. Preferably, the file system uses a write anywhere file system layout, the file operation that signals the reservation operation is a zero length write request, and the file operation that signals the reservation operation includes a parameter that specifies the file size. A number of blocks needed to be reserved to accommodate the file is computed. Preferably, computing the number of blocks needed to be reserved to accommodate the file includes determining a total number of direct and indirect blocks needed to accommodate the file size, and subtracting a total number of blocks already allocated for the file and a total number of cached unallocated blocks for the file from the total number of direct and indirect blocks needed to accommodate the file size. A number of unallocated blocks is reserved in the file system, with the number of reserved blocks equal to the number of blocks needed to be reserved to accommodate the file. Reserving the number of blocks preferably includes setting a flag in an inode for the file that indicates blocks have been reserved for the file, and incrementing a reserved block count in a file system information block by the number of blocks needed.

//

//

//

//

//

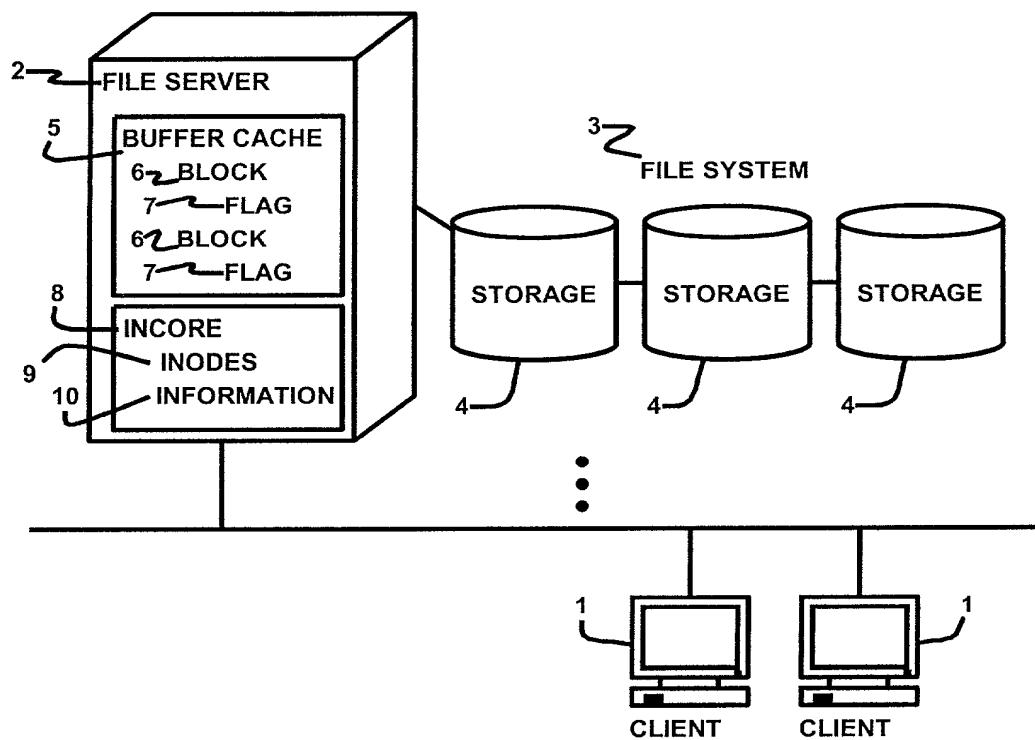


FIG. 1

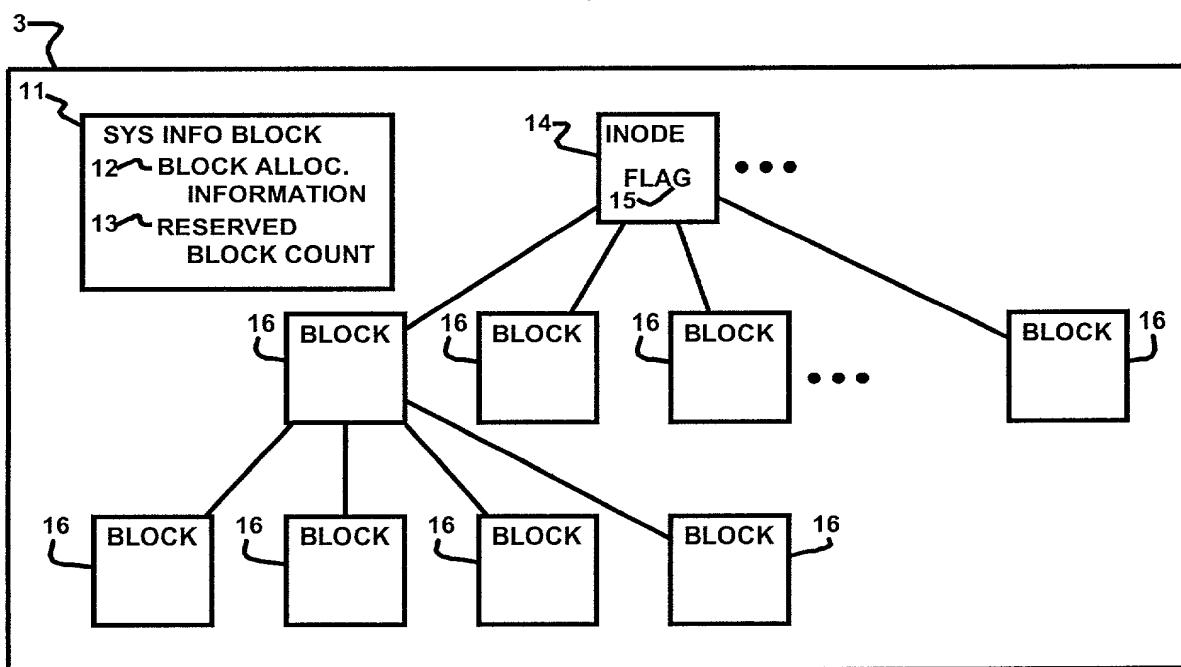
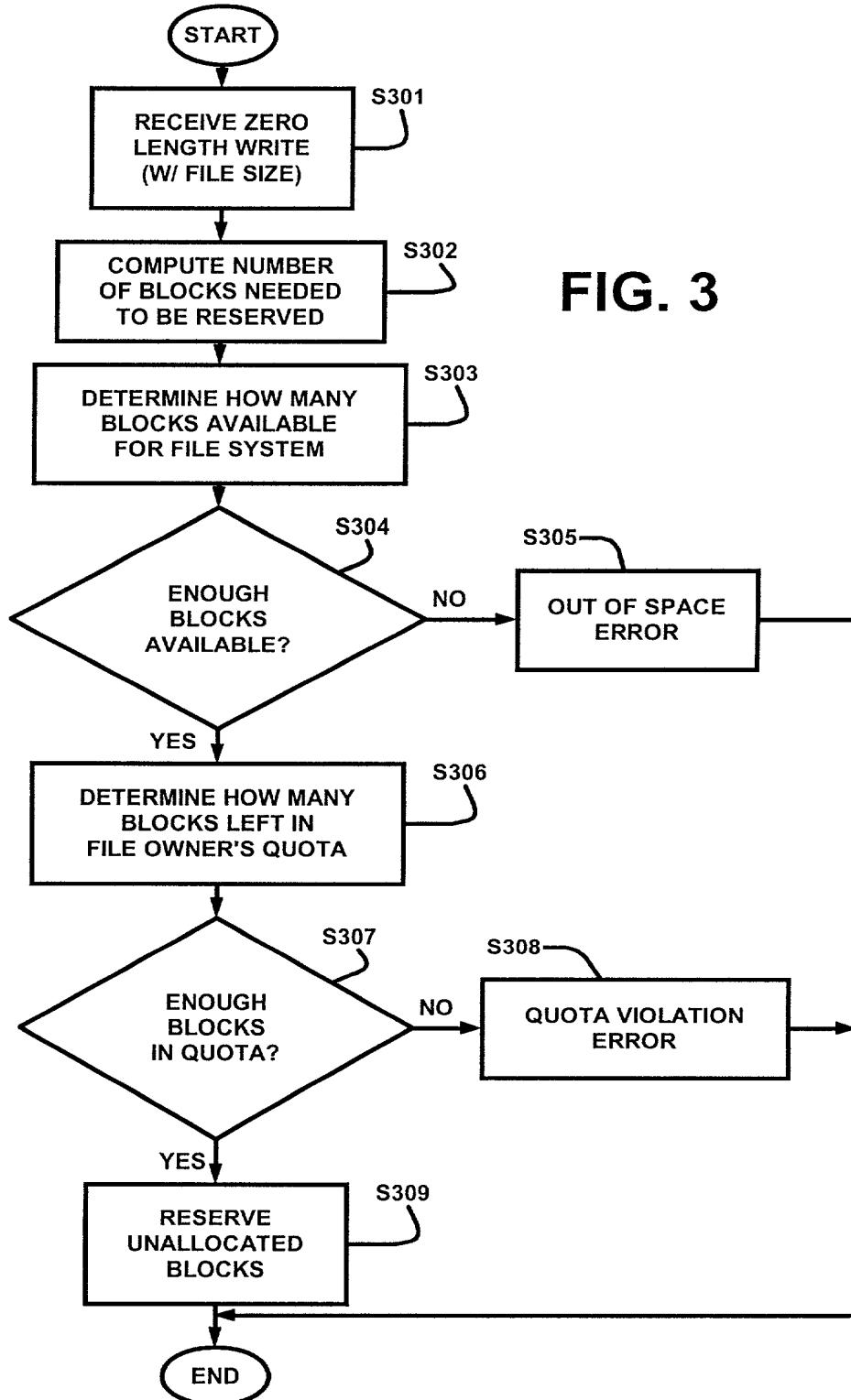
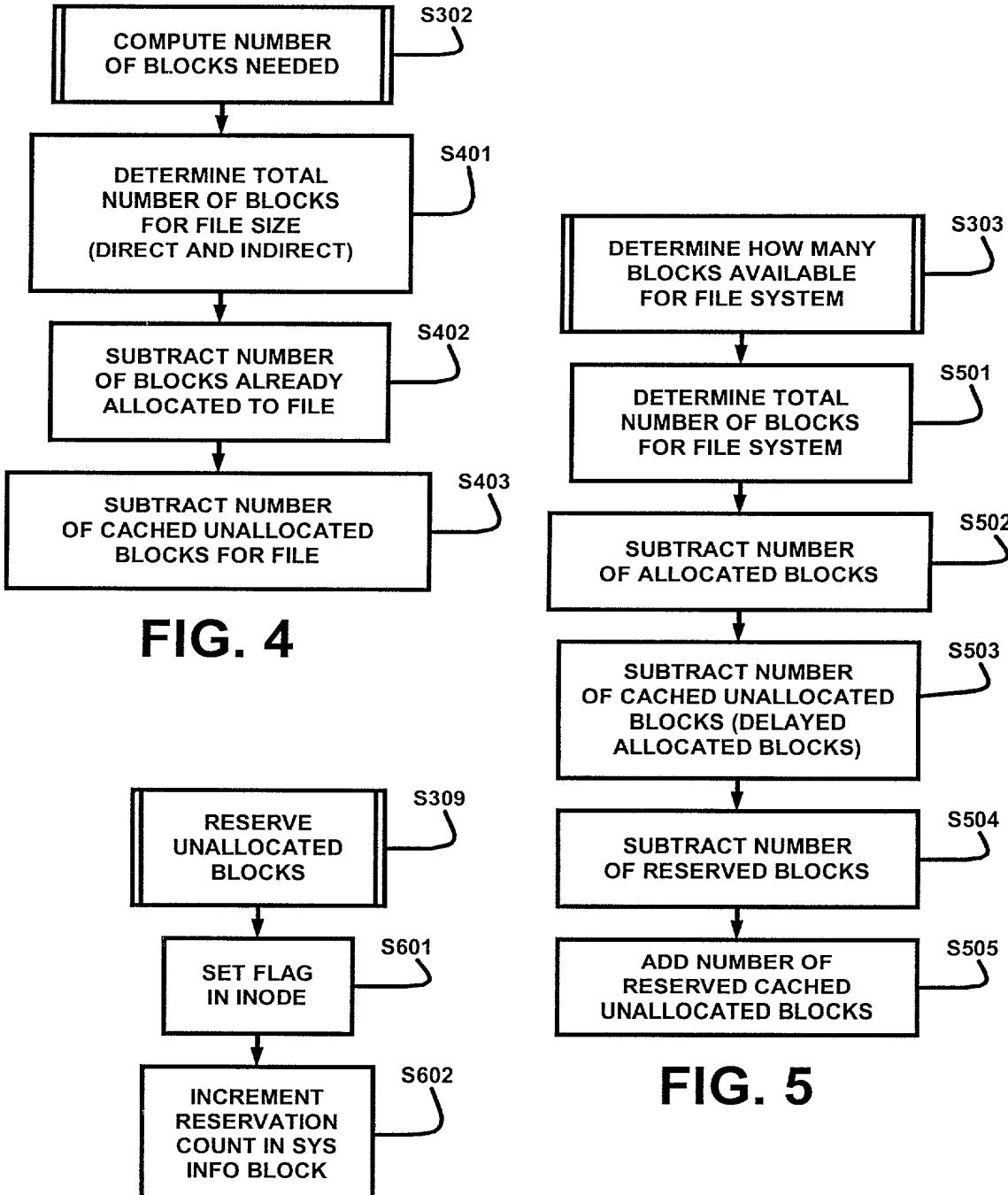
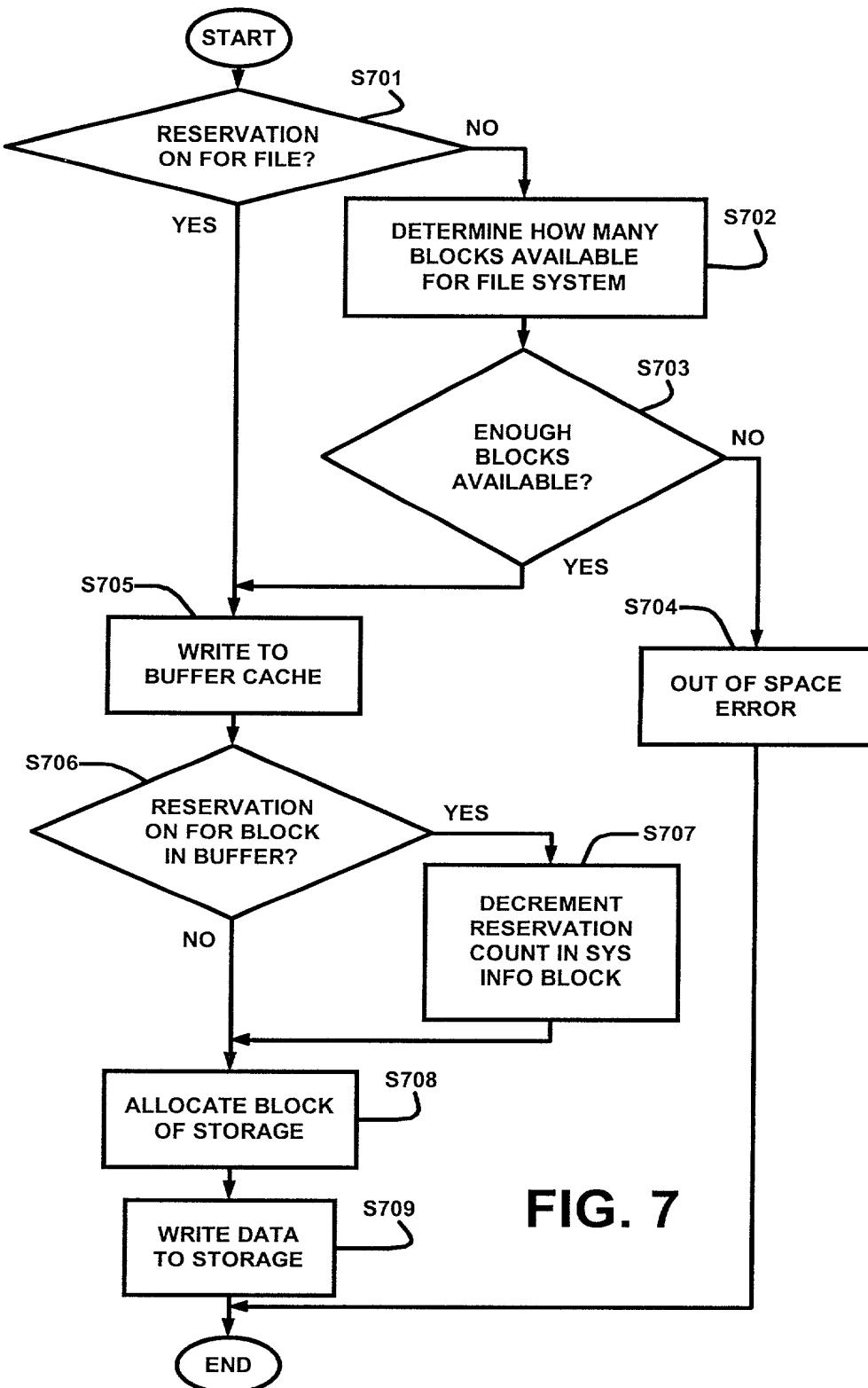


FIG. 2





**FIG. 7**

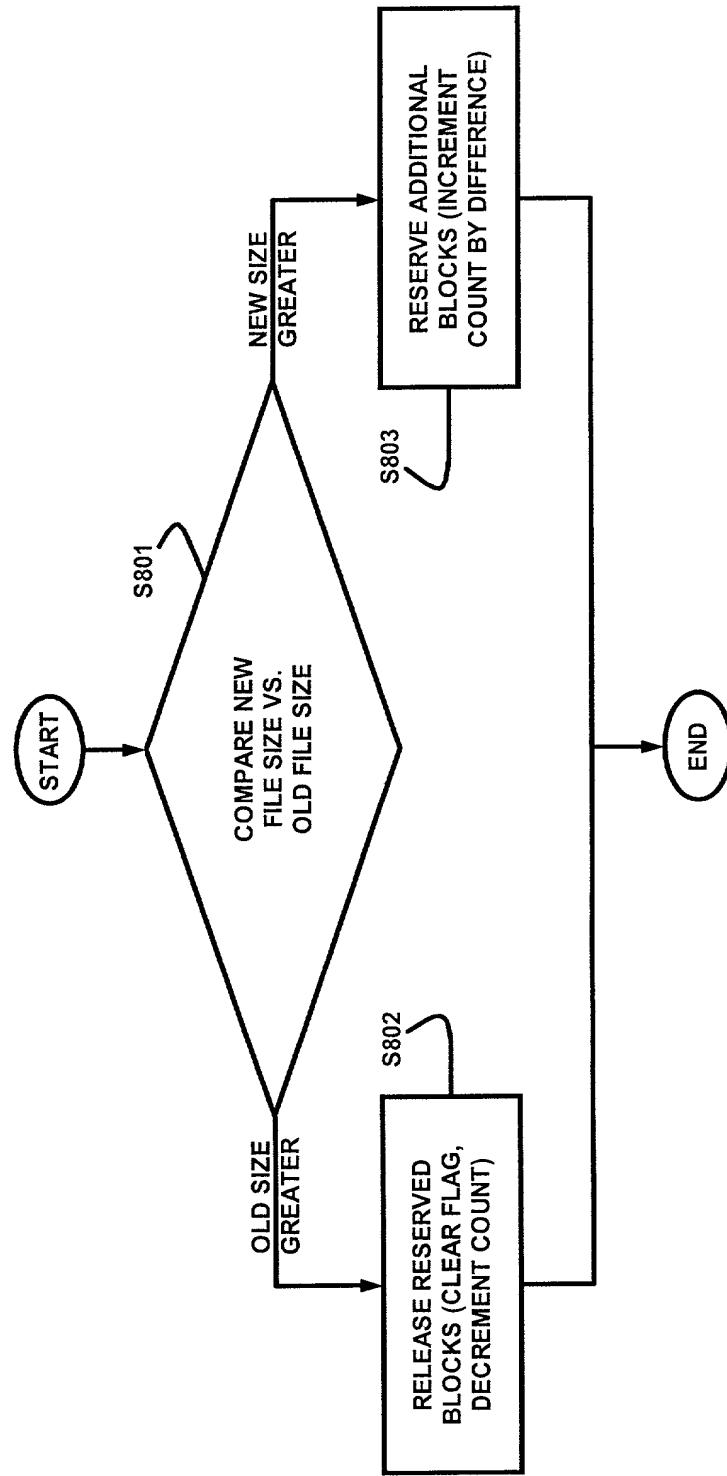


FIG. 8